

## پیش پردازش

### بخش ۱)

کد:

```
df = pd.read_csv('players.csv')

print(df.head)

#--1--
print("***** 1 *****")
print(df.head(1))
print(df.tail(1))
```

زمانی که دیتا ها را از فایل می خوانیم خود کتابخانه pandas داده ها را ایندکس گذاری می کند که از 0 شروع می شوند. فانکشن head() از سر فریم و tail() از انتهای فریم شروع می شوند که با مشخص کردن ایندکس می توانیم سطر مورد نظر را برگردانیم.

خروجی:

```
***** 1 *****
|  |  ID      Name      FullName  Age  ...  LBRating  CBRating  RBRating  GKRating
0  158023  L. Messi  Lionel Messi  33  ...      65        55        65        22

[1 rows x 90 columns]
|  |  ID      Name      FullName  ...  CBRating  RBRating  GKRating
19019  241493  S. Cartwright  Samuel Cartwright  ...      51        47        15

[1 rows x 90 columns]
```

### بخش ۲)

کد:

```
--2--
print("***** 2 *****")
print("number of all missing datas:", df.isna().sum().sum())
temp = df.isna().sum()/(len(df))*100
print("Column with lowest amount of missings contains {} % missings.".format(temp.min()))
print("Column with highest amount of missings contains {} % missings.".format(temp.max()))
print("columns contain missing values:", df.loc[:, df.isnull().any()].columns)
print("missing value checking:")
print(df.isnull())
print("rows contain missing values:")
print(df[df.isnull().any(axis=1)])
```

در این قسمت تعداد کل دیتا های از دست رفته، ستون هایی که بیشترین و کم ترین دیتا از دست رفته را شامل می شوند، اسم ستون هایی که شامل دیتا های از دست رفته هستند، دیتا فریم بصورتی که از دست رفته باشد این مقدار true در غیراینصورت false خواهد بود و سطر هایی که شامل دیتا از دست رفته هستند را مشخص کرده ایم.

خروجی:

```

***** 2 *****
number of all missing datas: 36492
Column with lowest amount of missings contains 0.0 % missings.
Column with highest amount of missings contains 94.08517350157729 % missings.
columns contain missing values: Index(['ClubPosition', 'ContractUntil', 'ClubNumber', 'NationalPosition',
    'NationalNumber'],
    dtype='object')
missing value checking:

```

	ID	Name	FullName	Age	...	LBRating	CBRating	RBRating	GKRating
0	False	False	False	False	...	False	False	False	False
1	False	False	False	False	...	False	False	False	False
2	False	False	False	False	...	False	False	False	False
3	False	False	False	False	...	False	False	False	False
4	False	False	False	False	...	False	False	False	False
...	...	...	...	...	...	...	...	...	...
19015	False	False	False	False	...	False	False	False	False
19016	False	False	False	False	...	False	False	False	False
19017	False	False	False	False	...	False	False	False	False
19018	False	False	False	False	...	False	False	False	False
19019	False	False	False	False	...	False	False	False	False

```

[19020 rows x 90 columns]
rows contain missing values:

```

	ID	Name	...	RBRating	GKRating
4	190871	Neymar Jr	...	65	23
6	208722	S. Man	...	69	23
11	212831	Alisson	...	33	91
14	200145	Casemiro	...	84	24
16	165153	K. Benzema	...	62	21
...	...	...	...	...	...
19015	257371	M. Nzongong	...	42	18
19016	259160	L. Bell	...	41	13
19017	259157	Y. Arai	...	44	17
19018	253763	R. Dinanga	...	34	16
19019	241493	S. Cartwright	...	47	15

```

[18118 rows x 90 columns]

```

بخش ۳)

کد:

```

#--3--
print("***** 3 *****")
print("Average Weights: ",df["Weight"].mean())
print("max Weight: ",df["Weight"].max())
print("min Weight: ",df["Weight"].min())

```

با کد df[Weight] ویژگی وزن را انتخاب و از روی آن میانگین، ماکزیموم و مینیموم را مشخص کرده ایم.

خروجی:

```
***** 3 *****  
Average Weights: 75.05241850683491  
max Weight: 110  
min Weight: 50
```

بخش ۴)

```
#-4--  
print("***** 4 *****")  
df = df.dropna(subset=['Nationality'])  
maxNumber = df['Nationality'].value_counts().head(1)  
minNumber = df['Nationality'].value_counts().tail(1)  
  
print("max: ", maxNumber)  
print("min: ", minNumber)
```

ویژگی ملیت را انتخاب و روی آن تابع `value_counts()` را اجرا کرده ایم. این تابع خودش تعداد تکرار هر element را محاسبه می کند. سپس `head(1)` ماکزیموم و `tail(1)` مینیموم را انتخاب می کنیم. البته در این بخش چندین کشور تعداد مینیموم داشتند که چون در صورت سوال خواسته نشده بود، در اینجا صرفاً آخرین را برگردانده ایم.

خروجی:

```
***** 4 *****  
max:  England    1706  
Name: Nationality, dtype: int64  
min:  Chad        1  
Name: Nationality, dtype: int64
```

بخش ۵)

```
#--5--
print("***** 5 *****")
dataFrame = df[df.Potential > 84]
FuturePlayers = dataFrame[dataFrame.Growth>4]
print(FuturePlayers)
```

در این بخش، در ابتدا یک فریم می‌سازیم که در آن potential بازیکنان بیشتر از 84 می‌باشد. سپس روی این فریم شرط دوم را چک و یک فریم جدید می‌سازیم.

خروجی:

```
***** 5 *****
```

	ID	Name	...	RBRating	GKRating
12	231747	K. Mbappé	...	66	21
45	231281	T. Alexander-Arnold	...	85	21
46	233049	J. Sancho	...	64	22
68	222492	L. Sané	...	62	20
70	228702	F. de Jong	...	84	21
...	...	...	...	...	...
10650	251873	Y. Demir	...	43	17
12472	247649	J. Branthwaite	...	61	18
12798	256781	L. Netz	...	63	18
14022	259419	T. Nakai	...	47	16
14290	258315	B. Arrey-Mbi	...	60	17

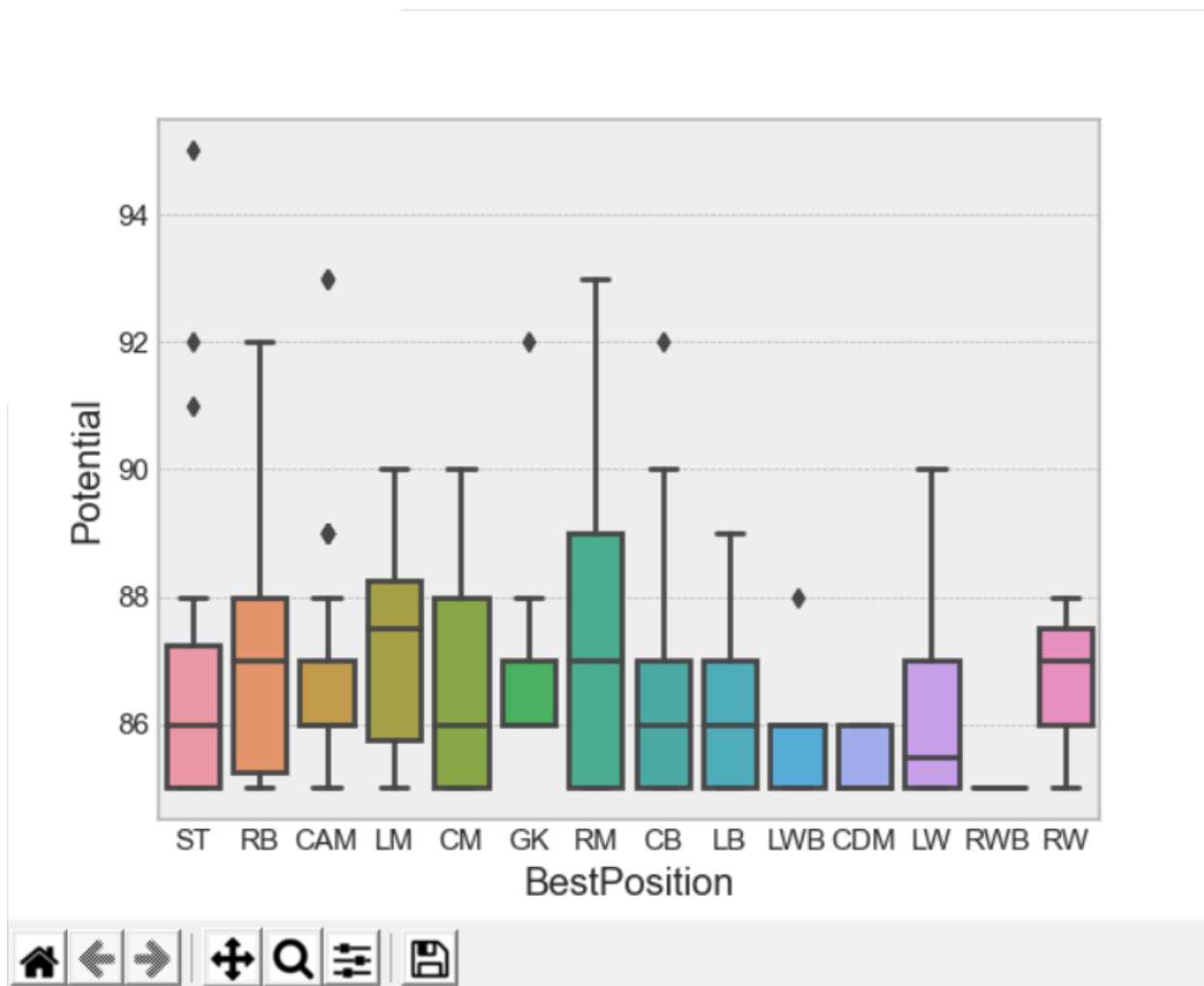
بخش ۶)

کد:

```
print("***** 6 *****")
sns.set_theme()
style.use('bmh')
sns.boxplot(x="BestPosition", y="Potential", data=FuturePlayers)
plt.show()
```

از فریم دیتا ای که از قبل ساخته بودیم یک نمودار رسم می‌کنیم که x آن برابر ستون BestPosition و ستون Potential را نمودار y ها در نظر می‌گیریم.

Figure 1



بخش ۷)

```
#--7--
print("***** 7 *****")
maxNumber = FuturePlayers['Club'].value_counts().head(1)
print("max: ", maxNumber)
```

در بخش روی فریم دیتا ای که از قبل ایجاد کرده بودیم، ویژگی باشگاه را انتخاب و بیشترین element تکرار شده را برمی گردانیم.

خروجی:

```
***** 7 *****
max: Sporting CP    10
Name: Club, dtype: int64
```

بخش ۸)

```
--8--
print("***** 8 *****")
print("total value of players: ", FuturePlayers[FuturePlayers.Club == "Chelsea"].ValueEUR.sum())
```

خروجی:

```
***** 8 *****
total value of players: 293900000
```

بخش ۹)

کد:

```
69  --9--
70  print("***** 9 *****")
71  df = df.dropna(subset=['ContractUntil'])
72  ContractUntil2021 = df[(df['ContractUntil'] == 2021)]
73  res = ContractUntil2021[ContractUntil2021.NationalTeam == "Not in team"]
74  print("count: ", res[res.columns[0]].count())
```

خروجی:

```
***** 9 *****
count: 6727
```

بخش ۱۰)

کد:

```
#--10--
print("***** 10 *****")
dataFrame = df[df.FullName == "Mehdi Taremi"]
print(dataFrame[["FullName", "Positions", "ValueEUR", "Club"]])
```

خروجی:

```
***** 10 *****
|   |   | FullName Positions  ValueEUR   Club
1017 Mehdi Taremi    ST,CF  11500000  FC Porto
1113 Mehdi Taremi    ST,CF  11500000  FC Porto
```

مشخصات مهدی طارمی در دو سطر تکرار شده است. در صورت سوال خواسته نشده بود که داده های تکراری را حذف کنیم.

قوانین انجمنی

(۱)



weka.gui.GenericObjectEditor

weka.associations.Apriori

**About**

Class implementing an Apriori-type algorithm.

More

Capabilities

car False

classIndex -1

delta 0.05

doNotCheckCapabilities False

lowerBoundMinSupport 0.2

metricType Confidence

minMetric 0.9

numRules 10

outputItemSets False

removeAllMissingCols False

significanceLevel -1.0

treatZeroAsMissing False

upperBoundMinSupport 1.0

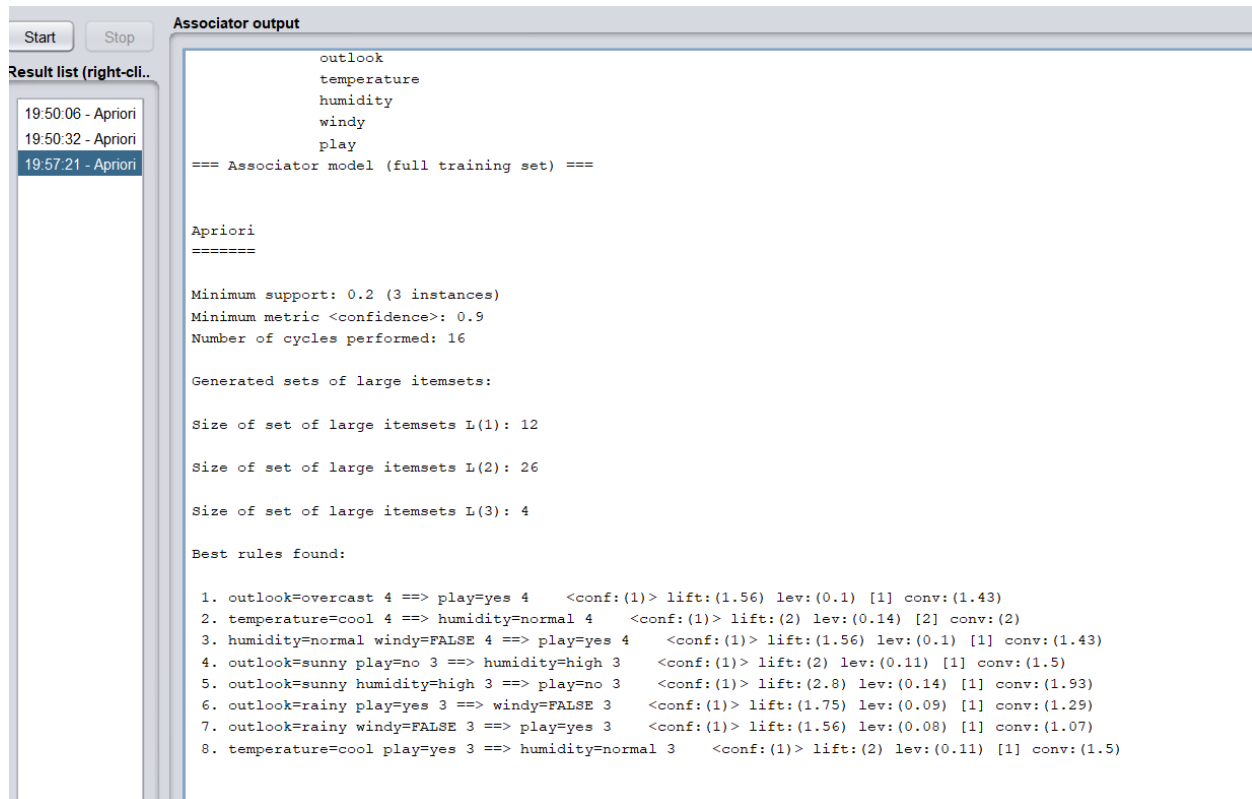
verbose False

Open... Save... OK Cancel

در این قسمت پارامتر های مختلفی وجود دارد که می توانیم تغییر دهیم از مهم ترین پارامتر ها لیمیت های پشتیبانی هستند، بخصوص آستانه پایین که در این الگوریتم بسیار مهم است. همینطور بخش مهم دیگر بخش

متریک ها هستند که در این قسمت ۴ متریک مختلف وجود دارد که می توانیم انتخاب و آستانه آن را تعیین کنیم که Confidence یکی از این حالات می باشد.

در ابتدا لیمیت تعداد قوانین را زیاد کردیم. دیدیم که تعداد ۳۳۶ قانون انجمنی محاسبه شده است.



```
outlook
temperature
humidity
windy
play

=== Associator model (full training set) ===

Apriori
=====

Minimum support: 0.2 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 26

Size of set of large itemsets L(3): 4

Best rules found:

1. outlook=overcast 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4    <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3    <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3    <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3    <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
```

حال مقدار آستانه پشتیبانی را به ۰,۲ افزایش می دهیم.

```

19:50:32 - Apriori
19:57:21 - Apriori
20:06:09 - Apriori
20:06:26 - Apriori
20:20:12 - Apriori

play
=== Associator model (full training set) ===

Apriori
=====

Minimum support: 0.2 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 26

Size of set of large itemsets L(3): 4

Best rules found:

1. outlook=overcast 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4    <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3    <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3    <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3    <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)

```

میبینیم که در هر مرحله تعداد زیادی از آیتم ها حذف شدند و همینطور تنها ۸ قانون انجمنی محاسبه شد. این قوانین قوی و بسیار مفید هستند زیرا هم ساپورت آن ها بالا است، یعنی تعداد تکرار آن ها زیاد است و هم میزان confidence برابر ۱ است که یعنی همیشه همراه یکدیگر بوده اند.

این مقدار را به ۰,۴ افزایش می دهیم می بینیم که ۶ آیتم تکی و ۲ آیتم دوتایی باقی ماند و هیچ قانون انجمنی حاصل نشد.

```

result list (right-click)
=====

Scheme:      weka.associations.Apriori -N 10000 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.4 -S -1.0 -c -1
Relation:    weather.symbolic
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play

=== Associator model (full training set) ===

Apriori
=====

Minimum support: 0.4 (6 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 12

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 2

Best rules found:

```

حال مقدار آستانه confidence را به ۰,۱ کاهش می دهیم.

```
19:50:06 - Apriori
19:50:32 - Apriori
19:57:21 - Apriori
20:06:09 - Apriori
20:06:26 - Apriori
20:20:12 - Apriori
20:22:00 - Apriori
20:33:07 - Apriori

Scheme:      weka.associations.Apriori -N 10000 -T 0 -C 0.1 -D 0.05 -U 1.0 -M 0.4 -S -1.0 -c -1
Relation:    weather.symbolic
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play

=== Associator model (full training set) ===

Apriori
=====

Minimum support: 0.4 (6 instances)
Minimum metric <confidence>: 0.1
Number of cycles performed: 12

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 2

Best rules found:

1. humidity=normal 7 ==> play=yes 6    <conf:(0.86)> lift:(1.33) lev:(0.11) [1] conv:(1.25)
2. windy=FALSE 8 ==> play=yes 6    <conf:(0.75)> lift:(1.17) lev:(0.06) [0] conv:(0.95)
3. play=yes 9 ==> humidity=normal 6    <conf:(0.67)> lift:(1.33) lev:(0.11) [1] conv:(1.13)
4. play=yes 9 ==> windy=FALSE 6    <conf:(0.67)> lift:(1.17) lev:(0.06) [0] conv:(0.96)
```

می بینیم که ۴ قانون حاصل شد.

۲ و ۳

در ابتدا فایل را میخوانیم. آدرس فایل برای سیستم خودم است و نیاز به تغییر دارد.

```
public class Main {

    public static void main(String[] args) throws Exception{

        String dataset = "C:\\Program Files\\Weka-3-8-5\\data\\supermarket.arff";
        DataSource source = new DataSource(dataset);
        Instances data = source.getDataSet();
    }
}
```

یک متغیر increment میسازیم که مقدارش طبق صورت سوال یک صدم اضافه شود و ساپورت های خواسته شده را ایجاد کند.

```

int counter = 0;
for (counter = 0; counter < 11; counter++) {
    Double num = Double.valueOf(counter)/100 + 0.05;
    // num = round(num,1);
    String increment = "" + num;
    String[] options = { "-N", "100000", "-C", increment, "-M", "0.1", "-S", "0.8"};

```

دو متغیر sTime و eTime تعریف می‌کنیم که با استفاده از آن‌ها زمان سیستم را در لحظه شروع و انتهای الگوریتم می‌گیریم تا بتوانیم زمان اجرا را برای هر دو الگوریتم محاسبه کنیم.

```

long sTime = System.currentTimeMillis();
Apriori apriori_model = new Apriori();
apriori_model.buildAssociations(data);
apriori_model.setOptions(options);
apriori_model.buildAssociations(data);
long eTime = System.currentTimeMillis();

long AprioryTime = (eTime - sTime);

sTime = System.currentTimeMillis();
FPGrowth fpgrowth_model = new FPGrowth();
//build model
fpgrowth_model.buildAssociations(data);
fpgrowth_model.setOptions(options);
fpgrowth_model.buildAssociations(data);
eTime = System.currentTimeMillis();

long FPGrowthTime = (eTime - sTime);

```

```

System.out.println("Min Support : " + increment);
System.out.println("Apriori time : " + AprioryTime + "milliseconds");
System.out.println("FP Growth time : " + FPGrowthTime + "milliseconds");
System.out.println("=====");

```

در آخر مدل‌ها را ایجاد و خروجی را print می‌کنیم.

خروجی:

```
Min Support : 0.05
Apriori time :18781milliseconds
FP Growth time :1870milliseconds
=====
Min Support : 0.060000000000000005
Apriori time :17180milliseconds
FP Growth time :1739milliseconds
=====
Min Support : 0.07
Apriori time :17418milliseconds
FP Growth time :1503milliseconds
=====
Min Support : 0.08
Apriori time :19468milliseconds
FP Growth time :1048milliseconds
=====
Min Support : 0.09
Apriori time :18154milliseconds
FP Growth time :1017milliseconds
```

```
=====
Min Support : 0.1
Apriori time :17666milliseconds
FP Growth time :946milliseconds
=====
Min Support : 0.11
Apriori time :16095milliseconds
FP Growth time :920milliseconds
=====
Min Support : 0.120000000000000001
Apriori time :17413milliseconds
FP Growth time :1129milliseconds
=====
Min Support : 0.13
Apriori time :17567milliseconds
FP Growth time :901milliseconds
=====
Min Support : 0.14
Apriori time :16274milliseconds
FP Growth time :888milliseconds
=====
Min Support : 0.150000000000000002
Apriori time :13540milliseconds
FP Growth time :719milliseconds
=====

Process finished with exit code 0
```